The Traveling Salesman Problem

Definition: A **complete graph** K_N is a graph with N vertices and an edge between every two vertices.

The Traveling Salesman Problem

Definition: A **complete graph** K_N is a graph with N vertices and an edge between every two vertices.

Definition: A **Hamilton circuit** is a circuit that uses every vertex of a graph once.

The Traveling Salesman Problem

Definition: A **complete graph** K_N is a graph with N vertices and an edge between every two vertices.

Definition: A **Hamilton circuit** is a circuit that uses every vertex of a graph once.

Definition: A **weighted graph** is a graph in which each edge is assigned a *weight* (representing the time, distance, or cost of traversing that edge).

The Traveling Salesman Problem

Definition: A **complete graph** K_N is a graph with N vertices and an edge between every two vertices.

Definition: A **Hamilton circuit** is a circuit that uses every vertex of a graph once.

Definition: A **weighted graph** is a graph in which each edge is assigned a *weight* (representing the time, distance, or cost of traversing that edge).

Definition: The **Traveling Salesman Problem** is the problem of finding a **minimum-weight Hamilton circuit** in K_N.

The Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at.

The Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at.

2. Walk to its nearest neighbor (i.e., along the shortest possible edge). (If there is a tie, break it randomly.)

The Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at.

2. Walk to its nearest neighbor (i.e., along the shortest possible edge). (If there is a tie, break it randomly.)

3. At each stage in your tour, walk to the nearest neighbor that you have not already visited.

The Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at.
2. Walk to its nearest neighbor (i.e., along the shortest possible edge). (If there is a tie, break it randomly.)
3. At each stage in your tour, walk to the nearest neighbor that you have not already visited.
4. When you have visited all vertices, return to the starting vertex.

The Repetitive Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at, and use the Nearest-Neighbor Algorithm to find a Hamilton circuit.

The Repetitive Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at, and use the Nearest-Neighbor Algorithm to find a Hamilton circuit.
2. Repeat Step 1 for every possible starting vertex. You will have a total of N Hamilton circuits.

The Repetitive Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at, and use the Nearest-Neighbor Algorithm to find a Hamilton circuit.
2. Repeat Step 1 for every possible starting vertex. You will have a total of N Hamilton circuits.
3. Select the cheapest one.

The Repetitive Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at, and use the Nearest-Neighbor Algorithm to find a Hamilton circuit.
2. Repeat Step 1 for every possible starting vertex. You will have a total of N Hamilton circuits.
3. Select the cheapest one.

- Usually, there is no way to know in advance which reference vertex will work the best.

The Repetitive Nearest-Neighbor Algorithm

1. Pick a reference vertex to start at, and use the Nearest-Neighbor Algorithm to find a Hamilton circuit.
2. Repeat Step 1 for every possible starting vertex. You will have a total of N Hamilton circuits.
3. Select the cheapest one.

- Usually, there is no way to know in advance which reference vertex will work the best.
- Once you find a Hamilton circuit, you can start your tour anywhere you want.

Example: Willy's Tour of Australia

Ref. vertex	Hamilton circuit	Weight	
AD	AD,ML,HO,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,PE,AL,AD	18543	
AL	AL,PE,BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,AL	19795	
AS	AS,UL,BM,KU,DA,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,AS	18459	
BT	BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,ML,HO,CN,SY,AL,PE,BT	22113	
BM	BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,PE,AL,BM	19148	
CS	CS,MK,BT,SY,CN,ML,HO,AD,AS,UL,BM,KU,DA,MI,PE,AL,CS	22936	
CN	CN,SY,ML,HO,AD,AS,UL,BM,KU,DA,MI,CS,MK,BT,AL,PE,CN	21149	
DA	DA,KU,BM,UL,AS,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,DA	18543	
HO	HO,ML,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,AL,PE,HO	20141	
KU	KU,DA,AS,UL,BM,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,KU	18785	
MK	MK,CS,MI,AS,UL,BM,KU,DA,AD,ML,HO,CN,SY,BT,AL,PE,MK	23255	
ML	ML,HO,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,AL,PE,ML	20141	
MI	MI,AS,UL,BM,KU,DA,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,MI	20877	
PE	PE,AL,BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,PE	19148	
SY	SY,CN,ML,HO,AD,AS,UL,BM,KU,DA,MI,CS,MK,BT,AL,PE,SY	21049	(NNA)
UL	UL,AS,MI,CS,MK,BT,SY,CN,ML,HO,AD,BM,KU,DA,PE,AL,UL	20763	

Example: Willy's Tour of Australia

Ref. vertex	Hamilton circuit	Weight	
AD	AD,ML,HO,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,PE,AL,AD	18543	
AL	AL,PE,BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,AL	19795	
AS	AS,UL,BM,KU,DA,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,AS	18459	**(best)**
BT	BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,ML,HO,CN,SY,AL,PE,BT	22113	
BM	BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,PE,AL,BM	19148	
CS	CS,MK,BT,SY,CN,ML,HO,AD,AS,UL,BM,KU,DA,MI,PE,AL,CS	22936	
CN	CN,SY,ML,HO,AD,AS,UL,BM,KU,DA,MI,CS,MK,BT,AL,PE,CN	21149	
DA	DA,KU,BM,UL,AS,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,DA	18543	
HO	HO,ML,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,AL,PE,HO	20141	
KU	KU,DA,AS,UL,BM,MI,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,KU	18785	
MK	MK,CS,MI,AS,UL,BM,KU,DA,AD,ML,HO,CN,SY,BT,AL,PE,MK	23255	
ML	ML,HO,CN,SY,BT,MK,CS,MI,AS,UL,BM,KU,DA,AD,AL,PE,ML	20141	
MI	MI,AS,UL,BM,KU,DA,CS,MK,BT,SY,CN,ML,HO,AD,AL,PE,MI	20877	
PE	PE,AL,BM,KU,DA,AS,UL,AD,ML,HO,CN,SY,BT,MK,CS,MI,PE	19148	
SY	SY,CN,ML,HO,AD,AS,UL,BM,KU,DA,MI,CS,MK,BT,AL,PE,SY	21049	(NNA)
UL	UL,AS,MI,CS,MK,BT,SY,CN,ML,HO,AD,BM,KU,DA,PE,AL,UL	20763	

The Repetitive Nearest-Neighbor Algorithm

Using Alice Springs (AS) as the reference vertex yields the best result:

AS $\rightarrow$ UL $\rightarrow$ BM $\rightarrow$ KU $\rightarrow$ DA $\rightarrow$ MI $\rightarrow$ CS $\rightarrow$ MK $\rightarrow$ BT $\rightarrow$ SY $\rightarrow$ CN $\rightarrow$ ML $\rightarrow$ HO $\rightarrow$ AD $\rightarrow$ AL $\rightarrow$ PE $\rightarrow$ AS

The Repetitive Nearest-Neighbor Algorithm

Using Alice Springs (AS) as the reference vertex yields the best result:

AS $\to$ UL $\to$ BM $\to$ KU $\to$ DA $\to$ MI $\to$ CS $\to$ MK $\to$ BT $\to$ SY $\to$ CN $\to$ ML $\to$ HO $\to$ AD $\to$ AL $\to$ PE $\to$ AS

Remember: Willy can still start anywhere he wants!
For instance,

SY $\to$ CN $\to$ ML $\to$ HO $\to$ AD $\to$ AL $\to$ PE $\to$ AS $\to$ UL $\to$ BM $\to$ KU $\to$ DA $\to$ MI $\to$ CS $\to$ MK $\to$ BT $\to$ SY

represents the same Hamilton circuit.

The Cheapest-Link Algorithm

Idea: **Start in the middle.**

The Cheapest-Link Algorithm

Idea: **Start in the middle.**

- Add the cheapest available edge to your tour.
  (If there is a tie, break it randomly.)

The Cheapest-Link Algorithm

Idea: **Start in the middle.**

- Add the cheapest available edge to your tour.
  (If there is a tie, break it randomly.)
- Repeat until you have a Hamilton circuit.

The Cheapest-Link Algorithm

Idea: **Start in the middle.**

- Add the cheapest available edge to your tour.
  (If there is a tie, break it randomly.)
- Repeat until you have a Hamilton circuit.
- Make sure you add exactly two edges at each vertex.
- Don't close the circuit until all vertices are in it.

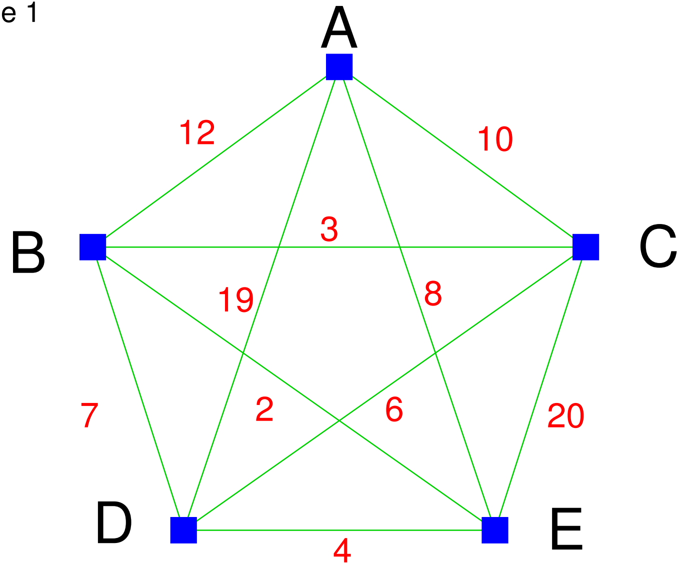
The Cheapest-Link Algorithm

Idea: **Start in the middle.**

- Add the cheapest available edge to your tour.
  (If there is a tie, break it randomly.)
- Repeat until you have a Hamilton circuit.
- Make sure you add exactly two edges at each vertex.
- Don't close the circuit until all vertices are in it.

This is called the **Cheapest-Link Algorithm, or CLA**.
Here is an example.

Example 1



Results of Example 1

- Output of RNNA: **BEDCAB** (weight **34**)
- Output of CLA: **ACBEDA** (weight **38**)

- In this example, RNNA produces a better result.

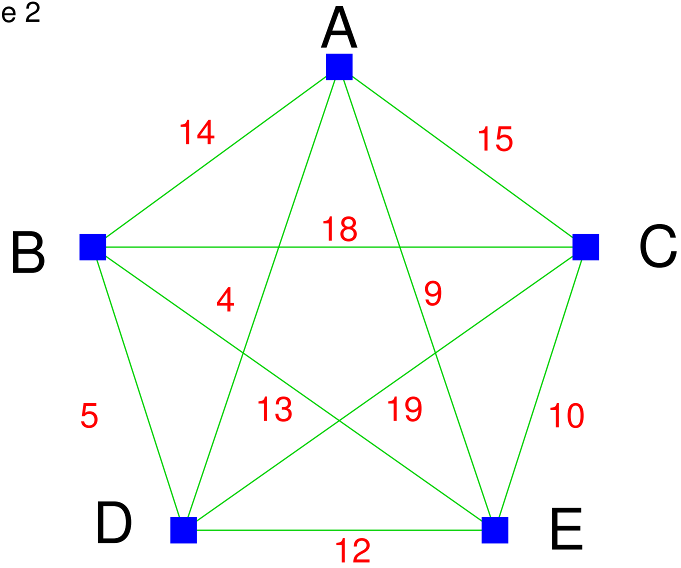
Results of Example 1

- Output of RNNA: **BEDCAB** (weight **34**)
- Output of CLA: **ACBEDA** (weight **38**)

- In this example, RNNA produces a better result.

- In fact, neither of these Hamilton circuits is optimal – the optimal one is **EACBDE** (weight **32**).

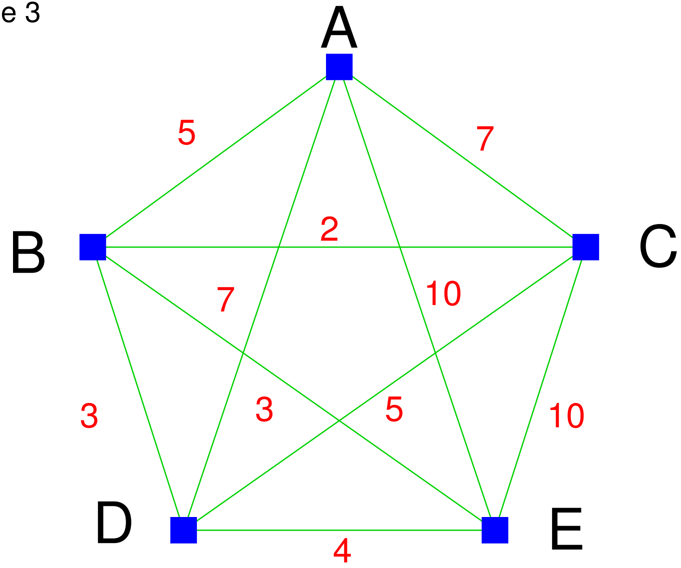
Example 2



Results of Example 2

- RNNA and CLA both output **DAECBD** (weight **46**)
- This happens to be an optimal Hamilton circuit.

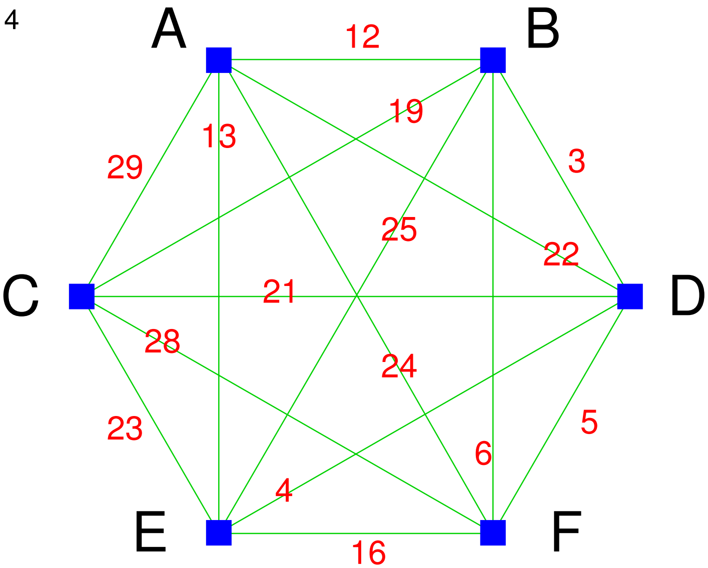
Example 3



Results of Example 3

- Here, the output of both the CLA and the RNNA may depend on how you break ties. (There's no way to know in advance.)

Example 4



Distance table for Example 4

	A	B	C	D	E	F
A		12	29	22	13	24
B	12		19	3	25	6
C	29	19		21	23	28
D	22	3	21		4	5
E	13	25	23	4		16
F	24	6	28	5	16	

Results of Example 4

- Output of RNNA: **FDBAECF** (weight **84**)
- Output of CLA: **ACFBDEA** (weight **83**)

- In this example, CLA produces a better result.

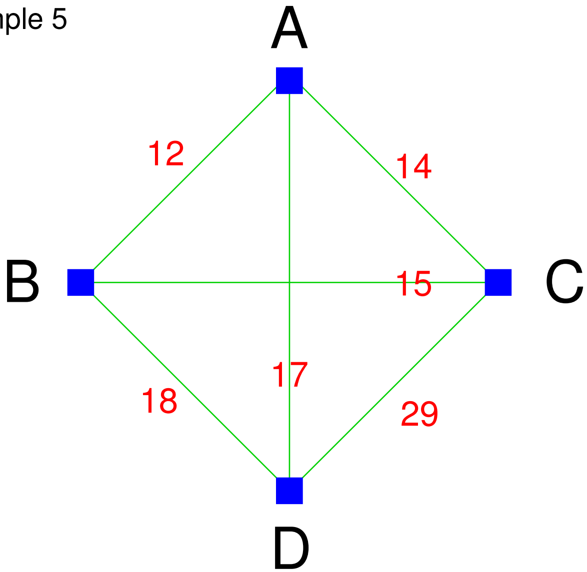
Results of Example 4

- Output of RNNA: **FDBAECF** (weight **84**)
- Output of CLA: **ACFBDEA** (weight **83**)

- In this example, CLA produces a better result.

- Neither of these Hamilton circuits is optimal – the optimal one is **FBCAEDF** (weight **76**).

Example 5



Results of Example 5

Algorithm	Output	Weight
NNA (A)	**ABCDA**	$12 + 15 + 29 + 17 =$ **73**
NNA (B)	**BACDB**	$12 + 14 + 29 + 18 =$ **73**
NNA (C)	CABDC	= BACDB
NNA (D)	DABCD	= ABCDA
CLA	ABCDA	

Results of Example 5

Algorithm	**Output**	**Weight**
NNA (A)	**ABCDA**	$12 + 15 + 29 + 17 =$ **73**
NNA (B)	**BACDB**	$12 + 14 + 29 + 18 =$ **73**
NNA (C)	CABDC	= BACDB
NNA (D)	DABCD	= ABCDA
CLA	ABCDA	

- The only other Hamilton circuit in K_4 is **ACBDA**, which has weight $14 + 15 + 18 + 17 =$ **64**.

Results of Example 5

Algorithm	Output	Weight
NNA (A)	**ABCDA**	$12+15+29+17=$ **73**
NNA (B)	**BACDB**	$12+14+29+18=$ **73**
NNA (C)	CABDC	= BACDB
NNA (D)	DABCD	= ABCDA
CLA	ABCDA	

- The only other Hamilton circuit in K_4 is **ACBDA**, which has weight $14+15+18+17=$ **64**.
- So both RNNA and CLA give **worst possible answers!**

The Bad News

There is no known algorithm to solve the TSP that is both optimal and efficient.

The Bad News

There is no known algorithm to solve the TSP that is both optimal and efficient.

- Brute-force is optimal but not efficient.
- NNA, RNNA, and CLA are all efficient but not optimal.